APPENDIX A

```
#!/usr/local/bin/perl

# Copyright (c) 2001
# LSI Logic, Inc.
#
# Andy Rankin
# Ph: (970) 206-5107
# email: arankin@lsil.com
#
# sp2vlog.pl
#
# Usage: sp2vlog.pl -s spfile [-d dfile] [-h]
#
#         -s <spfile>    Spice transistor level file to be
translated (required)
#         -d <dfile>     Data file from vlog2sp.pl containing port
types (defaults to basename.data)
#         -h             prints this message
#

# ADD CLEANER TO THIS FILE TO GET + SIGNS REMOVED
# ADD SOMETHING TO VLOG2SP TO MAKE SPICE COMPATIBLE LINE LENGTHS

################
# Set Defaults #
################

$tech="gflx";
$subckt_lead_string="x";

################################
# Parse Command Line Arguments #
################################

require "newgetopt.pl";
&NGetOpt("s:s","d:s","h");

if ($opt_h) { &usage; }
if ($opt_s) {
     $spfile=$opt_s;
}
else {
     print("\nERROR: Must specify spice file\n");
     &usage;
}

$basename=`basename $spfile .sp`;
chop($basename);
```

```
if ($opt_d) { $dfile=$opt_d; }
else { $dfile=$basename.".data"; }

###############################
# Open spice and data files #
###############################

open(INFILE,"$spfile") || die("Unable to open $spfile for
reading: $!\n\n");
@infile=<INFILE>;
close(INFILE);

open(INFILE,"$dfile") || die("Unable to open $dfile for reading:
$!\n\n");
@dfile=<INFILE>;
close(INFILE);

foreach (@infile) {
    @pin_list=();
    tr/A-Z/a-z/;
    if (/^\s*.subckt\s/) {
        # If the line is a spice subckt definition, it is the
same as a module definition in
        # verilog. Split the line to extract the subcircuit
name and pins:
        chop;
        ($foo,$module,@subckt_pins)=split(/\s+/);
        @module_pins=&remove_busses(@subckt_pins); # remove
bus notation from subcircuit pins
        $pinlist=join(", ",@module_pins);
        print ("module $module \( $pinlist \)\;\n"); # print
the verilog module definition line

        # For each pin on the module, determine the port type
from the data file:
        foreach $pin (@module_pins) {
            foreach $port (@dfile) {
                if ($port =~ /$module $pin (\S+)/) {
$port_type=$1; last; }
                else { $port_type=""; }
            }
            @indices=&consolidate_bus($pin,@subckt_pins); #
If it's a bus, add bus notation

            # Print the input/output definitions with bus
notation as necessary:
            if ($port_type) {
                if (@indices) { print("$port_type
[$indices[$#indices]:$indices[0]] $pin\;\n"); }
                else { print("$port_type $pin\;\n"); }
            }
```

```
      }

           # Print wire definitions for all the internal signals:
           foreach (@dfile) {
                 if (/$module (\S+) wire/) { print("wire $1\;\n");
}
           }

      }
      elsif (/^\s*.ends\s*/) {
           print("endmodule\n\n");
      }
      elsif (/^\s*x/) {
           s/^\s+//;
           s/\s+$//;
           @signal_names=split(/\s+/);
           $instance_name=shift(@signal_names);      # Shift the
instance name from front of line
           $instance_name=~s/^$subckt_lead_string//; # Remove the
X from the
           $subckt_name=pop(@signal_names);          # Pop the
module name from the end of line
           @signal_names=&clean_signals(@signal_names);

           #grep the spice file for the subcircuit definition,
and extract the pin names:
           ($foo,$foo2,@subckt_pins)=split(/\s+/,`grep -i
".SUBCKT $subckt_name " $spfile`);

           # Verify that the number of signals in the spice
subcircuit call matches the number of
           # pins in the spice subcircuit definition:
           if ($#signal_names != $#subckt_pins) {
                 die("ERROR: $instance_name $subckt_name number of
pins do not match\n\n");
           }

           # Pair up the signal names in the spice subcircuit
call with the pin names
           # on the spice subcircuit definition:
           for($i=0;$i<=$#signal_names;$i++) {
                 $subckt_pins[$i]=~tr/A-Z/a-z/;

      push(@pin_list,".$subckt_pins[$i]\($signal_names[$i]\)");
           }

           # Join the list of signal/pin pairs and print it with
the module instantiation:
           $pin_string=join(", ",@pin_list);
           print("$subckt_name $instance_name ( $pin_string
)\;\n");
```

```
        }


    }

sub usage {

        # This subroutine prints usage information

        print("\nusage: sp2vlog.pl -s spfile [-d dfile] [-h]\n\n");
        print("\t-s <spfile>\tspice transistor level file to be
translated (required)\n");
        print("\t-d <dfile>\tdata file from vlog2sp.pl containing
port types (defaults to basename.data)\n");
        print("\t-h\t\tprints this message\n");
        die("\n");
}

sub consolidate_bus {

        # This subroutine finds the indices for a given pin, sorts
them, and returns them.

        my($pin,@pins) = @_;
        my(@indices);

        foreach (@pins) {
                if (/$pin\[(\d+)\]/) {
                        push(@indices,$1);
                }
        }
        @indices=sort(@indices);

}

sub remove_busses {

        # This subroutine removes bus notation from the pins on the
spice subcircuit line.
        # It includes each bus as one pin in the pinlist. It
returns the pin list.

        my(@pinlist) = @_;
        my(@newpinlist);

        foreach (@pinlist) {
                s/\[\d+\]//g;
                if(&element_exists($_,@newpinlist)) {}
                else { push(@newpinlist,$_); }
        }
```

```
     return(@newpinlist);
}

sub element_exists {

     # This subroutine checks to see if an element exists in an
array.

     my($element,@array)=@_;

     foreach (@array) {
          if ($_ eq $element) { return(1); }
     }

     return(0);
}

sub clean_signals {

     # This subroutine converts vss and vdd signals to verilog
logic 0 and logic 1. It also
     # removes dummy pins.

     my(@signals) = @_;

     foreach (@signals) {
          if ($_ eq "vss") { $_ = "1'b0"; }
          if ($_ eq "vdd") { $_ = "1'b1"; }
          if (/^dummy/) { $_ = ""; }
     }
     @signals;
}
```

United States Patent & Trademark Office

Office of Initial Patent Examination – Scanning Division

SCANNED,# 24

Application deficiencies found during scanning:

☐ Page(s)__________of____________________________ were not present for scanning.
(Document title)

☐ Page(s)__________of____________________________ were not present for scanning.
(Document title)

* pages numbered 14-18 as part of specification are Appendix.

☐ ***Scanned copy is best available.***